ARTIST

FP7-317859

Advanced software-based seRvice provisioning and migraTIon of legacy Software

Deliverable D5.3.3

Technical Feasibility Tools

Editor(s):	Burak Karaboğa, Jesus Gorroñogoitia
Responsible Partner:	Atos
Status-Version:	Final – v1.0
Date:	15/03/2015
Distribution level (CO, PU):	PU

Project Number:	FP7-317859
Project Title:	ARTIST

Title of Deliverable:	D5.3.3 Technical Feasibility Tools
Due Date of Delivery to the EC:	30/03/2015

Workpackage responsible for the Deliverable:	WP5 – Modernization Assessment
Editor(s):	Atos - Burak Karaboğa, Jesus Gorroñogoitia
Contributor(s):	Atos – Burak Karaboğa, Jesus Gorroñogoitia Tecnalia – Zurik Corera, Juncal Alonso
Reviewer(s):	Sparxsystems – Konrad Wieland
Approved by:	All Partners
Recommended/mandatory readers:	WP6, WP7, WP8, WP9, WP11

Abstract:	These tools analyse the complexity and coupling of the source code of the legacy product, its data schema and database technologies, and create high level abstract models out of the legacy source code that address the concerns related to its migration into the desired target cloud platform, providing an estimation of the work (effort) that would be needed to transform the legacy product into that target platform.
Keyword List:	Technical feasibility, Complexity, Maintainability
Licensing information:	TFT is delivered under EPL license. The document itself is delivered as a description for the European Commission about the released software, so it is not public.

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
v0.1	10/03/15	First contributions for M30 release	ATOS
V0.2	13/03/15	Improvements	ATOS
V0.3	13/03/15	Merging Tecnalia's Contributions	ATOS/Tecnalia
V0.4	16/03/15	Wording corrections	ATOS
V1.0	30/03/15	Corrections and updates after internal review	ATOS

Table of Contents

Table of Contents 4
Table of Figures 5
Terms and abbreviations 6
Executive Summary 7
1 Introduction 9
1.1 About this deliverable 9
1.2 Document structure 9
2 TFT 11
2.1 Implementation 11
2.1.1 Functional Description 11
2.1.1.1 Fitting into overall ARTIST solution 11
2.1.2 Technical description 12
2.1.2.1 Prototype architecture 12
2.1.2.2 TFT-UI 14
2.1.2.3 Components description 15
2.1.2.4 Technical Specifications 16
2.2 Delivery and usage 18
2.2.1 Package information 18
2.2.2 Installation instructions 19
2.2.3 User manual 20
2.2.3.1 Migration goals view 21
2.2.3.2 Inventory view 22
2.2.3.3 Migration strategy selection dialog 23
2.2.3.4 Migration efforts view 24
2.2.4 Licensing information 26
2.2.5 Download 26
3 SCC 27
3.1 Implementation 27
3.1.1 Functional Description 27
3.1.1.1 Fitting into overall ARTIST solution 27
3.1.2 Technical description 28
3.1.2.1 Prototype Architecture 28
3.1.2.2 Components description 28
3.1.2.3 Technical specifications 29
3.2 Delivery and usage 29

3.2.1 Package information 29
3.2.2 Installation instructions 30
3.2.2.1 Requirements 31
3.2.3 User manual 31
3.2.4 Licensing information 32
3.2.5 Download 32
4 Interaction of TFT components 33
5 Conclusions 34
6 References 35
7 Appendix: TFT knowledge base (library of rules) 36
7.1 Migration Strategy Taxonomy 36
7.2 Migration strategies 38

Table of Figures

FIGURE 1 - ARTIST MIGRATION FEASIBILITY ASSESSMENT PACKAGE 12
FIGURE 2 - TFT ARCHITECTURE 13
FIGURE 3 - INVENTORY VIEW 14
FIGURE 4 - MIGRATION EFFORTS VIEW 14
FIGURE 5 - MIGRATION GOALS VIEW 15
FIGURE 6 - TFT PREFERENCES 15
FIGURE 7 - PACKAGE STRUCTURE OF TFT 18
FIGURE 8 - ECLIPSE WORKBENCH WITH TFT VIEWS 20
FIGURE 9 - OPENING THE MIGRATION GOALS VIEW 21
FIGURE 10 - MIGRATION GOALS VIEW 22
FIGURE 11 - OPENING A MODEL IN INVENTORY VIEW 22
FIGURE 12 - INVENTORY VIEW 23
FIGURE 13 - OPENING MIGRATION STRATEGY SELECTION DIALOG 23
FIGURE 14 - MIGRATION STRATEGY SELECTION DIALOG 24
FIGURE 15 - SOURCE PATHS SELECTIONS DIALOG 25
FIGURE 16 - MIGRATION EFFORTS VIEW 25
FIGURE 17 - EFFORT DETAILS DIALOG 26
FIGURE 18 - SCC HIGH LEVEL ARCHITECTURE 28
FIGURE 19 - PACKAGE STRUCTURE OF SCC 30
FIGURE 20 - SCC PROJECT 31
FIGURE 21 - TEST PACKAGE 31
FIGURE 22 - SCC TESTING 32
FIGURE 23 - INTERACTION OF TFT COMPONENTS 33

Terms and abbreviations

EC	European Commission
TFT	Technical Feasibility Tool
MAT	Maturity Assessment Tool
EMF	Eclipse Modelling Framework
UML	Unified Modelling Language
JAXB	Java Architecture for XML Binding
URL	Uniform Resource Locator
MPT	Methodology Process Tool
MUT	Model Understanding Tool
GML	Goal Modeling Language

Executive Summary

This document describes the Technical Feasibility Tool (TFT), which studies the feasibility of the legacy application's migration to cloud within the context of ARTIST pre-migration approach, along with detailing the Software Complexity Component (SCC) which plays a major role in this feasibility analysis.

The main motivation behind TFT is to assist the user to have a better understanding of the migration to cloud process and the efforts required to accomplish this process in the ARTIST pre-migration phase.

The purpose of this deliverable is to describe the modifications and new features developed and released in the second version of the Technical Feasibility Tool detailing its design and implementation and thus give an idea about the tool's usability, capabilities and potential.

This version (M30) of the Technical Feasibility Tool introduces some improvements and changes to the existing UI elements and backend components. Besides these updates, the tool's functionalities and capabilities are also extended with new visual elements.

Regarding the visual improvements, absolute efforts column has been added to the Migration Efforts View which estimates and shows the needed absolute effort values for each component by using the effort data provided by the user for a task. The view now also shows the overall averages at the bottom of the view to give the user a better idea of the big picture. The effort details dialog which can be accessed within the Migration Efforts View now provides the information on the estimated effort values for each individual migration strategy suggested for the selected component. The Migration Efforts View as also been updated due to the addition of the support for the new MAT report in GML format. In this version, TFT also contributes to Eclipse UI's perspectives and preference pages extension points defining its own Eclipse perspective and adding a preference page under the ARTIST category.

Regarding the implementation improvements, addition of suggestions to the application level component in the Inventory View, improving the performance of the Inventory View by moving the effort calculations to the Migration Efforts View's initialization (which reduces the time needed for the Inventory View to be opened), automatic estimations of absolute efforts upon user input in Migration Efforts View's Effort Details dialog, support for the usage of the new MAT report in GML format (which offers more detailed data on the migration goals) and finally the knowledge base of TFT has been improved to host even more rules covering more complex strategy scenarios. The final version of the SCC incorporates several novelties such as the inclusion of all the required atomic metrics to calculate the composed maintainability metric as well as the calculation of a new metrics related to the reusability of the legacy code. With respect to this last metric, this version of the SCC counts the number of reusable lines of code and the number of reusable classes in the original application. These new metrics contribute to increase the accuracy of the estimation for the migration strategy effort.

The document presents the missions, scopes, functional descriptions, technical approaches, download & installation instructions and user manuals of these components comprising the current version of the Technical Feasibility tools at M30. The deliverable focuses on improvements and extensions upon the initial version of TFT provided on M24 and it refers to deliverable D5.3.2 for unchanged features and details of the tool.

The next iteration of this deliverable is planned to be at M36 with increased capabilities and stability, a better integration and communication between TFT components and external dependencies of TFT (e.g. other ARTIST tools). The next version is planned to extend the existing knowledge base with new rules to handle more complex scenarios, make smarter migration strategy suggestions and estimate efforts needed for migrations more precisely.

1 Introduction

1.1 About this deliverable

The purpose of Technical Feasibility Tools (TFT) is to offer an early technical analysis of the migration of an existing non-Cloud compliant application, by conducting an analysis of the Maturity Assessment Tool's (MAT) report, and the high level component models of the application extracted by the Model Understanding Toolbox, in order to obtain a set of component maintainability metrics, migration strategy suggestions and migration effort estimations, which altogether offers a migration feasibility technical report. In the context of ARTIST, TFT assists the user in the pre-migration phase by displaying the analysis results and suggesting migration strategies to the users about their existing systems in a responsive way.

TFT is comprised of different components which have different roles in the analysis process of the migration candidate legacy application. These components are: component detector, strategy suggestor, software complexity component, effort estimator and TFT repository, last of which is not offered in this version. Component detector is responsible for analyzing the high level component model obtained from MUT and population of the component data model in the memory. Strategy suggestor examines this data model and migration goals obtained from MAT in order to suggest appropriate migration strategies for each component. Software complexity component (SCC) is responsible for the analysis of the source code, class model and the component model to calculate the complexity metrics for the whole application and each individual high level component. The effort estimator uses the complexity values detected by SCC and the migration strategies suggested by strategy suggestor to estimate the effort needed to migrate each component and also the whole application to the target platform.

This deliverable describes the technical and functional aspects, the role in the overall ARTIST solution, installation details and user manuals of TFT components in detail while also focusing on the new features and changes that has been implemented on them for the M30 prototype release.

1.2 Document structure

This document is divided into three main sections describing TFT in general, detailing the SCC component of TFT and lastly presenting the interaction between TFT components. The document is concluded followed by the references and the appendix.

This version of the deliverable focuses on the updates and new features with respect to the previous release, therefore the unchanged sections/subsections will refer to the old version. The first two sections are divided into subsections; Implementation and Delivery and Usage. Implementation sections focus on functional and technical aspects of the components in several subsections of their own. These subsections include information about functional aspects, fitting into overall ARTIST solution and technical aspects such as prototype architecture, components description and technical specifications. Delivery and Usage section focuses on what is delivered, how to download, install and use this deliverable. The package information subsection describes the package contents; the classes, resources and library files used by the component. Download and Installation Instructions sections explain how to download the prototype from the repository, how to install it as well as the prerequisites in detail. The User Manual section describes the Eclipse Views and other elements that the user can interact with that are offered by each component of TFT.

Interaction of TFT Components section focuses on the communication between TFT IDE and SCC components, describing the API offered by SCC and how TFT uses this API to calculate effort estimations for suggested migration strategies.

The Appendix section describes the taxonomy of migration tasks and lists the remarkable migration strategies offered by TFT in two subsections.

2 TFT

2.1 Implementation

2.1.1 Functional Description

The Technical Feasibility Tool (TFT), whose technical description is provided in [1], aims at supporting ARTIST users on the early technical assessment of the migration of a legacy application to the cloud. At this early stage (e.g. pre-migration phase in ARTIST Methodology), the ARTIST users need to be supported to evaluate the feasibility of the migration, attending its technical aspects, since even for a very simple legacy application, its migration to the cloud could be a complex process, that may require non negligible efforts and concrete expertise to be accomplished. Moreover, the support for decision making at this early pre-migration stage requires a detailed breakdown of the migration process into a set of technical tasks, not only to estimate their required efforts, but also to identify other resources needed to accomplish every task, including the selection of the appropriate technical expertise or even the detection of dependencies among tasks or other technical intricacies.

In this context, the role of this version of TFT prototype is to offer technical information about the legacy application itself, suggest technical strategies to migrate its components to the cloud and to offer migration effort estimations for each component marked for migration by analyzing the information in the migration goals of the MAT report, the high level view of the application provided by Model Understanding Tool (MUT), SCC's metric analysis as well as user preferences [3] and lastly generation of the TFT report for the consumption of Methodology Process Tool (MPT).

The prototype is responsive to user interactions such as, modifications on MUT's output, elimination of migration goals from MAT's report and re-assignment of migration strategy suggestions on components. These interactions trigger the analysis process of TFT and thus updating the migration strategy suggestions, complexity and effort estimations.

TFT may be considered as a high-level technical analysis for the whole legacy application. What MUT provides to TFT is a filtered, refined model of the legacy system and this output requires a certain analysis of structure to be created as well but this reveals what the system "is". TFT, on the other hand, aims to find out what the application "can" be or "should" be. This is why TFT's analysis is considered to be high-level.

2.1.1.1 Fitting into overall ARTIST solution

Next picture depicts the main TFT components and their dependencies and/or relationships with other ARTIST components. Note that not all TFT foreseen dependencies and interactions, as described in next figure, are implemented in M30 prototype. As commented in this section, current version of TFT integrates MAT reports and MUT high level models. For further details see [5] D5.3.2 Technical Feasibility Tools and [6] D6.4.1 ARTIST Integrated Architecture.

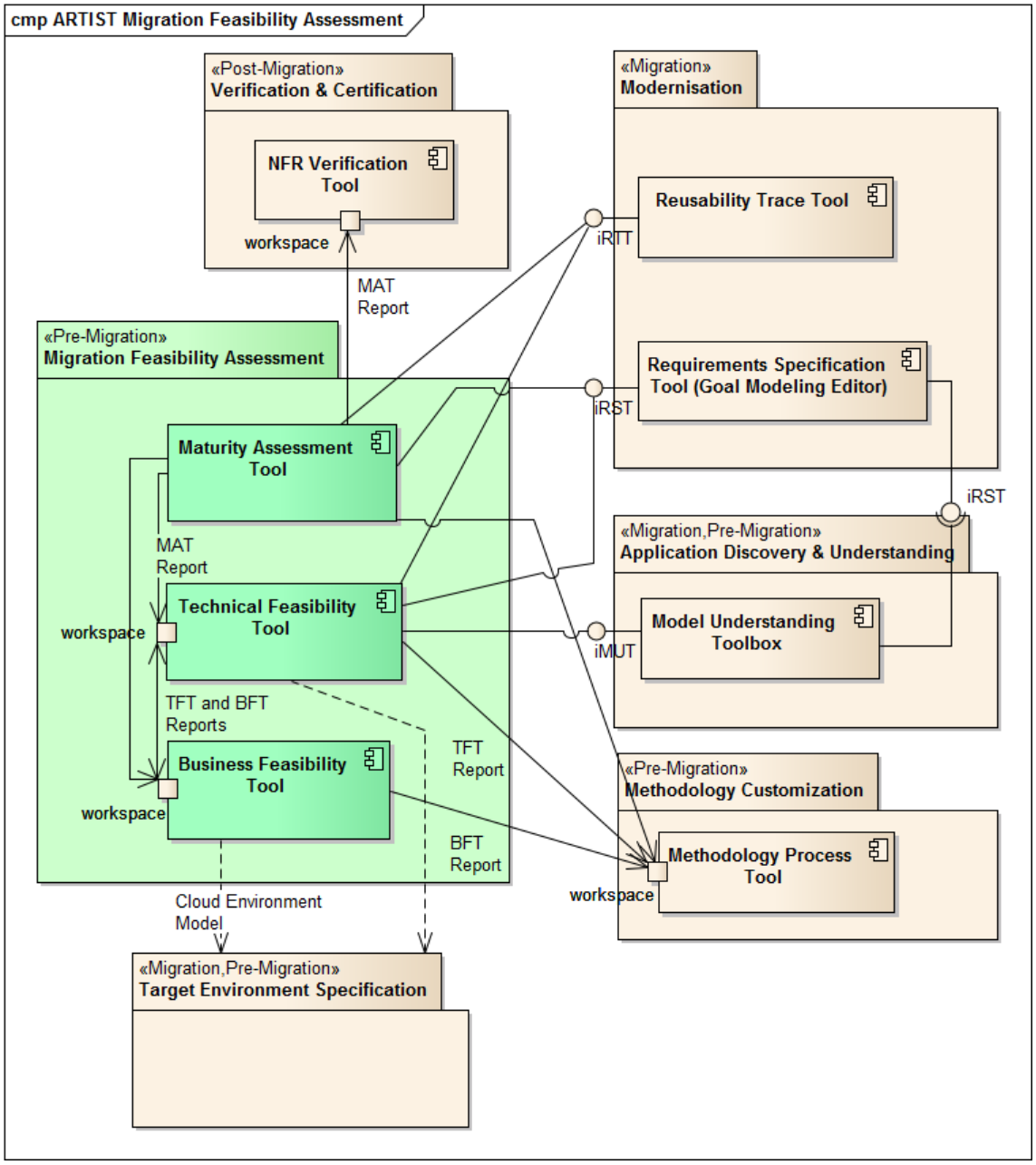


Figure 1 - ARTIST Migration Feasibility Assessment Package

2.1.2 Technical description

2.1.2.1 Prototype architecture

Next figure outlines TFT tool architecture design. The current implementation includes new versions of the following components: Software Complexity Component (SCC), TFT UI and Strategy Suggestor. This report describes the complete TFT architecture for the M30 prototype.

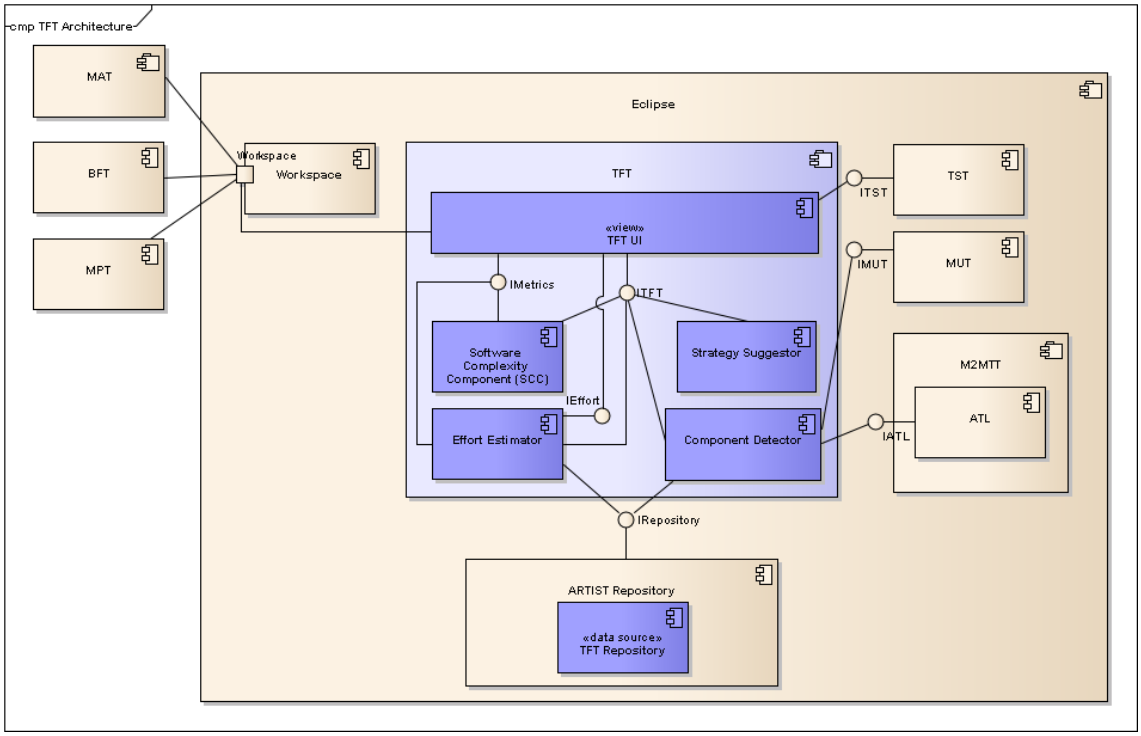


Figure 2 - TFT Architecture

TFT consists of a set of Eclipse elements (views, widgets, and wizards), a set of backend components and a set of external dependencies; all of which are described below.

TFT tool consists of:

- a set of Eclipse views and other widgets and wizards, which constitute the TFT UI and contributes to the Eclipse workbench through its exposed extension points. In this version (M30) of the prototype, the existing views, Inventory View, Migration Goals and Migration Efforts View have been improved both in their functions and visuals.
- a set of backend components, which provides some important TFT services, such as:
    - Component Detector
    - Software Complexity Component
    - Effort Estimator
    - Strategy Suggestor
    - TFT Repository (a bunch of TFT required artefacts stored in the ARTIST Repository)

  Software Complexity Component and Effort Estimator have been improved, the details of these improvements can be found in section 2.1.2.3. Strategy Suggestor has also been improved in this version, the details of which can also be found in section 2.1.2.3.
- a set of external dependencies, notably other ARTIST components and tools, accessed through well-defined interfaces:
    - Maturity Assessment Tool (MAT)
    - Business Feasibility Tool (BFT)
    - Target Specification Tool (TST)
    - Model Understanding Tool (MUT)
    - M'M Transformation Tool (M2MTT)
    - Reusability Analysis Tool (RAT)
    - ARTIST Repository
    - Methodology Framework Tool (MFT)

There has been no significant updates on the interaction between TFT and these external dependencies except a better analysis is now carried out for MUT's output, more information added on TFT report and TFT now supports the consumption of MAT's new report in GML (Goal Modeling Language) format to be used as the migration goals input.

Next sections describe in more details TFT plugin components and dependencies.

2.1.2.2 TFT-UI

TFT UI contributes to the Eclipse workbench, complementing the Eclipse ARTIST perspective. The following views have been improved or newly developed in this M30 TFT UI release (see [D5.3.2] for detailed description of TFT UI views included in M24 TFT release):

- Inventory of components View: M30 version of this view has been updated with small visual updates to present the user a better understanding of the view and some other non-visual updates have been done to improve the performance. Application component's migration strategy suggestions are now shown in this view which can also be modified like the rest of the components' migration strategy suggestions. The effort calculations for the migration strategies of the components are now moved to the initialization of the Migration Efforts View which reduced the time needed to open the Inventory View significantly.

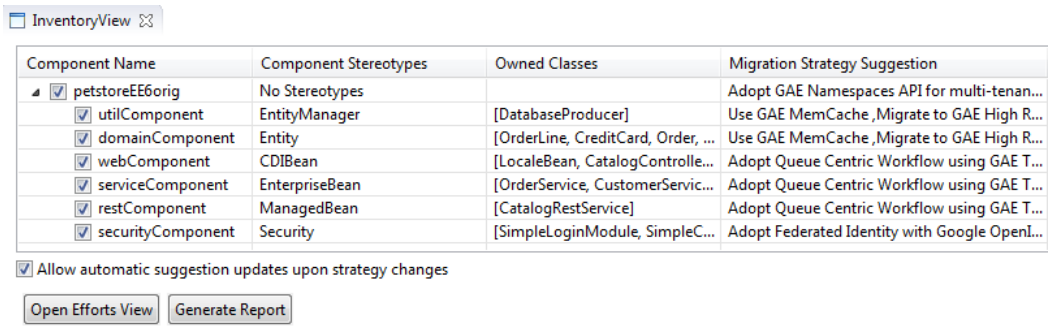


Figure 3 - Inventory view

- Migration Efforts View: The new version of this TFT view is improved to be easier to understand and to have more user interaction. The most significant change on this view is the addition of the Absolute Effort column which allows the user to set an absolute person/day effort value for a migration of a component. As in the Inventory View, this view has also been updated to show data for the application component as well. The view also shows the overall average effort values to the user.

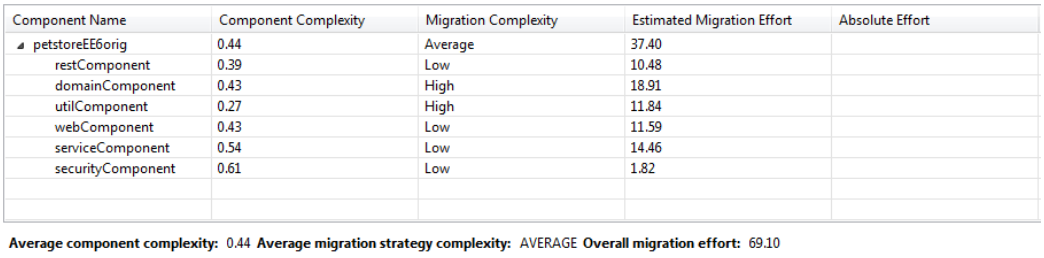


Figure 4 - Migration efforts view

- Migration Goals View: This version of the prototype's migration goals view now supports the new MAT report in GML format which introduces some visual updates to the view. In this version, the migration goals are shown under two main categories; Hard Goals and Soft Goals. New columns are also added to the view; Kind, Priority, Condition, Property and Threshold. Currently, the values in these columns are not utilized in suggestion process except property and condition values. The data on the other columns are planned to be utilized in the M36 version of the deliverable. Although the visuals are changed and new information on the goals is introduced, there is no change with respect to how this view is used. The user is still free to modify the target platform and the selection of the goals as in the previous version of the prototype.

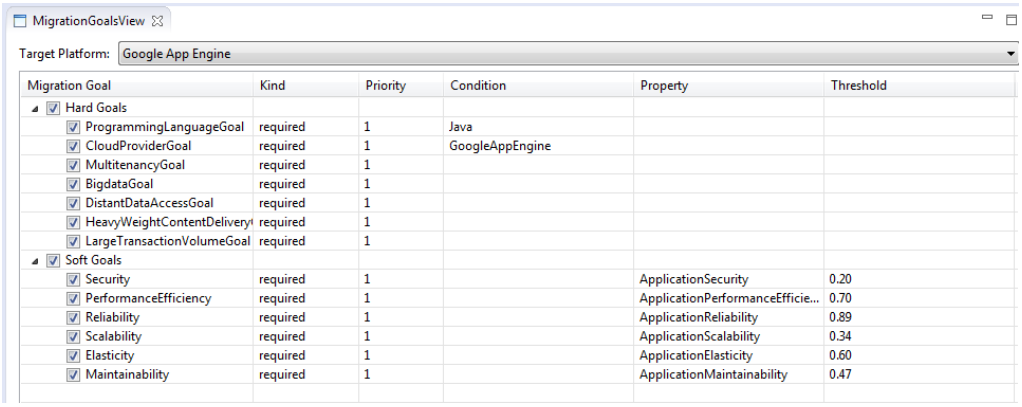

MigrationGoalsView
Target Platform: Google App Engine
Migration Goal | Kind | Priority | Condition | Property | Threshold
Hard Goals
ProgrammingLanguageGoal | required | 1 | Java
CloudProviderGoal | required | 1 | GoogleAppEngine
MultitenancyGoal | required | 1
BigdataGoal | required | 1
DistantDataAccessGoal | required | 1
HeavyWeightContentDelivery | required | 1
LargeTransactionVolumeGoal | required | 1
Soft Goals
Security | required | 1 | ApplicationSecurity | 0.20
PerformanceEfficiency | required | 1 | ApplicationPerformanceEfficie... | 0.70
Reliability | required | 1 | ApplicationReliability | 0.89
Scalability | required | 1 | ApplicationScalability | 0.34
Elasticity | required | 1 | ApplicationElasticity | 0.60
Maintainability | required | 1 | ApplicationMaintainability | 0.47

Figure 5 - Migration goals view

- TFT Perspective: M30 version of TFT introduces an Eclipse Perspective called TFT, contributing to the Eclipse UI's perspectives extension. This perspective defines a default layout for TFT which includes Migration Goals View, Inventory View and Migration Efforts View. Since all of these depend on user interaction to show data, the initial states of all these views contain a single line of text informing the user on how to activate the view. The user is free to move, close and resize the views.
- TFT Preferences: This version also adds a preference page for TFT under the parent ARTIST Preference Page, on which the user is offered the option to set the default location path of the TFT report to be stored in.

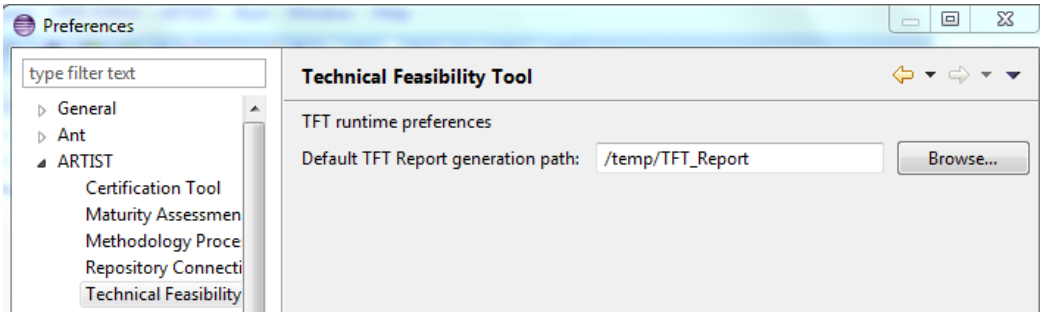


Figure 6 - TFT preferences

- See [5] D5.3.2 - Technical Feasibility Tools for other views which belong to TFT-UI.

2.1.2.3 Components description

The following TFT backend components provide business logic support to the TFT UI:

Component Detector: See [5] D5.3.2 Technical Feasibility Tools.

Software Complexity Component: This component has been updated with several changes and improvements in this version of the prototype, including new metrics calculations. These updates are detailed in section 3 of the document.

Strategy Suggestor: The new version of this component has been improved to make even more smart suggestions by adding the information of the migration goal selections of the user in the Migration Goals View to the process of strategy suggestion. The component has also been updated with new migration strategies thus offering the user more options for migration of components in the application, some of which are explained in detail in the Appendix: TFT knowledge base (library of rules) section.

Effort Estimator: See [5] D5.3.2 Technical Feasibility Tools.

TFT Repository: This component is not offered in this version of the prototype but planned to be available with the next deliverable in M36. This component will store historical data and heuristics required to estimate efforts. The M30 TFT implementation is local, that is, this content is shipped within TFT plugin upon installation from ARTIST update site. Incoming version will rely on the ARTIST repository to store and retrieve this information.

2.1.2.4 Technical Specifications

TFT, as an Eclipse plugin, is based on Eclipse RCP libraries and written in Java. TFT requires Java 6, a minimum Eclipse version of 4.3 (Kepler), and plugins EMF-Ecore, EMF-Query, Papyrus, UML2 installed.

User Interface

M30 version of TFT does not include a change in technology used within the user interface except the addition of the TFT Perspective and TFT Preference Page contributing to the Eclipse UI's perspective and preference page extensions respectively. For functional changes and improvements on TFT UI, see section 2.1.2.2.

For technical specifications of TFT UI, see [5] D5.3.2 Technical Feasibility Tools

Back-End

Although there is no introduction of new technologies in this version (M30), there are some new improvements on TFT with regards to M24 version of the prototype. For the technologies used in the back-end of TFT see [5] D5.3.2 Technical Feasibility Tools

TFT now supports the usage of MAT's new report in GML format which required significant changes in the parsing of the MAT input. The NFRVT (Non-functional Requirements Verification Tools) plugins are added as dependencies to TFT to be able to parse the new MAT report by using the methods provided in these plugins. NFRVT resources defining the ARTIST types are also added to the source folder structure of TFT in order to interpret the data gained by parsing MAT report and use this data on the Migration Goals View.

The knowledge base of TFT has been enhanced with more rule and migration strategy suggestion additions for the target platforms; AWS, Windows Azure and Google App Engine, improving the existing complex suggestion scenarios not only for the components of the application, but for the application itself as well. An example to a migration strategy suggestion scenario from M30 version of TFT would be:

```
rule "IsStandardComponent"
 when
   $component : Component ( !applicationLevelComponent )
 then
   //SUPER RULE - DO NOTHING
end

rule "IsDataTypeComponent" extends "IsStandardComponent"
 when
   eval ($component.hasStereotype(Category.DATA.getPossibleStereotypes()))
 then
   //SUPER RULE - DO NOTHING
End

rule "Data_NOSQL" extends "IsDataTypeComponent" agenda-group "GOOGLE_APP_ENGINE"

    when
        eval(goals.requiresSoftGoal("Availability") ||
        goals.requiresSoftGoal("Scalability") || goals.requiresHardGoal("Multitenancy")
        || goals.requiresSoftGoal("NO-SQL"))
    then
        MigrationStrategy strategy =
        MigrationStrategyContainer.INSTANCE.getMigrationStrategy("HRD");
        strategy.setComplexity(Complexity.HIGH);
        strategy.setSuggested(true);
        $component.addMigrationStrategy(strategy);
end

rule "Data_RDBMS" extends "IsDataTypeComponent" agenda-group  "GAE"

   when
        eval( goals.hasSoftGoal("RDBMSmultitenant") &&
        goal.isItemChecked("RDBMSmultitenant")
   then
        MigrationStrategy strategy =
        MigrationStrategyContainer.INSTANCE.getMigrationStrategy("CLOUD_SQL");
        strategy.setComplexity(Complexity.LOW);
        strategy.setSuggested(true);
        $component.addMigrationStrategy(strategy);
end
```

For a given component which is not marked as an application level component, the first rule, IsStandardComponent returns true. Then a second rule IsDataTypeComponent, which extends the first one, gets fired and if this component has one of the stereotypes in the DATA category applied on it, the rule returns true. The third and fourth rules extend the IsDataTypeComponent rule so both of them get fired. The third rule checks if any of the goals namelyAvailability, Scalability, Multitenancy or NO-SQL goals are required. This is done by invoking the *requiresHardGoal* or *requiresSoftGoa*l methods in the *MigrationGoals* class both of which check if the goal is present in the migration goals and is marked as checked by the user. If the evaluation returns true, the HRD strategy is applied to the component and the complexity is set to HIGH. If not, and there is a soft goal named RDBMSmultitenant which is also checked, then the migration strategy is set to CLOUD_SQL and complexity to LOW. The future work plan includes improving the code by changing the String values of the goals by a more modular and easier to maintain data structure as well as enhancing and improving these rules even more to be more comprehensive, making use of all the information we have on the legacy components in a smart way to decide which migration strategy and which level of complexity should be assigned to the component.

2.2 Delivery and usage

2.2.1 Package information

Delivery package consists of the below folder structure.

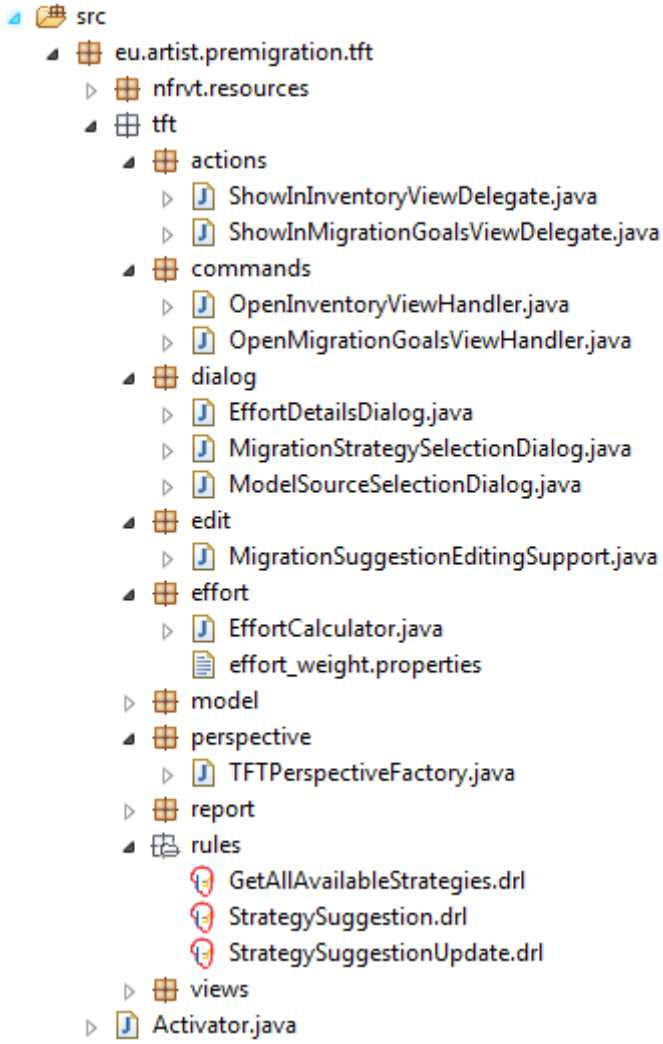


Figure 7 - Package structure of TFT

src: See [5] D5.3.2 - Technical Feasibility Tools.

eu.artist.tft.nfrvt.resources: Contains the necessary classes and enumerations implemented in the context of WP11, needed for interpreting the new MAT report in GML format

eu.artist.tft.tft.actions: See [5] D5.3.2 - Technical Feasibility Tools.

commands: See [5] D5.3.2 - Technical Feasibility Tools.

dialog: See [5] D5.3.2 – Technical Feasibility Tools.

edit: See [5] D5.3.2 – Technical Feasibility Tools.

effort: See [5] D5.3.2 – Technical Feasibility Tools.

model: This package contains Java models of UML components: *Component.java*, Migration Goals: *MigrationGoals.java and MigrationGoalsContainer.java.* The package also contains the enumeration *ProgrammingLanguage.java* which is used throughout TFT. The sub-package query contains *ComponentModelQuery.java* class. This class has the necessary methods for querying the UML Component Diagrams. The migration strategy sub-package contains *Category.java*, *Complexity.java*, *MigrationStrategies.java*, *MigrationStrategy.java*, *MigrationStrategyContainer.java*, *TargetPlatform.java*, *MigrationStrategies.xml*, and *MigrationStrategies.xsd*. The classes in this sub-package are java class mappings of the migration strategies which are defined in the *MigrationStrategies.xml*.

perspective: This package contains the *TFTPerspectiveFactory.java* defining the default layout of the Eclipse perspective of TFT.

report: This package has only one class called *ReportGenerator.java* which handles the report generation function of TFT rules**:** See [5] D5.3.2 - Technical Feasibility Tools.

views: See [5] D5.3.2 - Technical Feasibility Tools.

models: See [5] D5.3.2 - Technical Feasibility Tools.

profiles: See [5] D5.3.2 - Technical Feasibility Tools.

lib: See [5] D5.3.2 - Technical Feasibility Tools.

META-INF: See [5] D5.3.2 - Technical Feasibility Tools.

plugin.xml: See [5] D5.3.2 - Technical Feasibility Tools.

2.2.2 Installation instructions

TFT plugin will be available for installation using Eclipse's Install New Software feature via ARTIST's update site in the future. Now, it can be downloaded and installed manually. Eclipse 4.2 (Juno) is required for installation but Eclipse 4.3 (Kepler) is recommended.

Installation Procedure of Required Plugins

EMF-Ecore, EMF-Query, Papyrus, UML2, NFRVT plugins are required to be installed. All these plugins except NFRVT may be installed by following the installation procedure below:

1. On the Help menu of Eclipse, click Install New Software.
2. From the "Work with" dropdown list select Kepler or Juno depending on your Eclipse version. If you don't have these entries in the list continue on from step 2a otherwise continue from step 3.
    a. Click Add button at the right side of the Work with dropdown list.
    b. Write a name of your choice for the Kepler/Juno repo.
    c. Write http.download.eclipse.org/releases/kepler or http://download.eclipse.org/releases/juno to the Location field and click Ok.
3. Select the plugin(s) you want to install (EMF Core, EMF Query, UML2 and Papyrus UML) click next and follow the instructions on screen.

4. Restart Eclipse.

Installation Procedure of NFRVT plugins

Please refer to the readme section here:

https://github.com/artist-project/ARTIST/tree/master/binary/NFRVT

TFT Installation Procedure

1. Download eu.artist.premigration.tft.jar and eu.artist.premigration.tft.scc.jar files from ARTIST repository.
2. Copy these jar files to Eclipse's dropins folder.
3. Restart Eclipse

2.2.3 User manual

A general view of the Eclipse Workbench with TFT views and views that interact with TFT open is below.

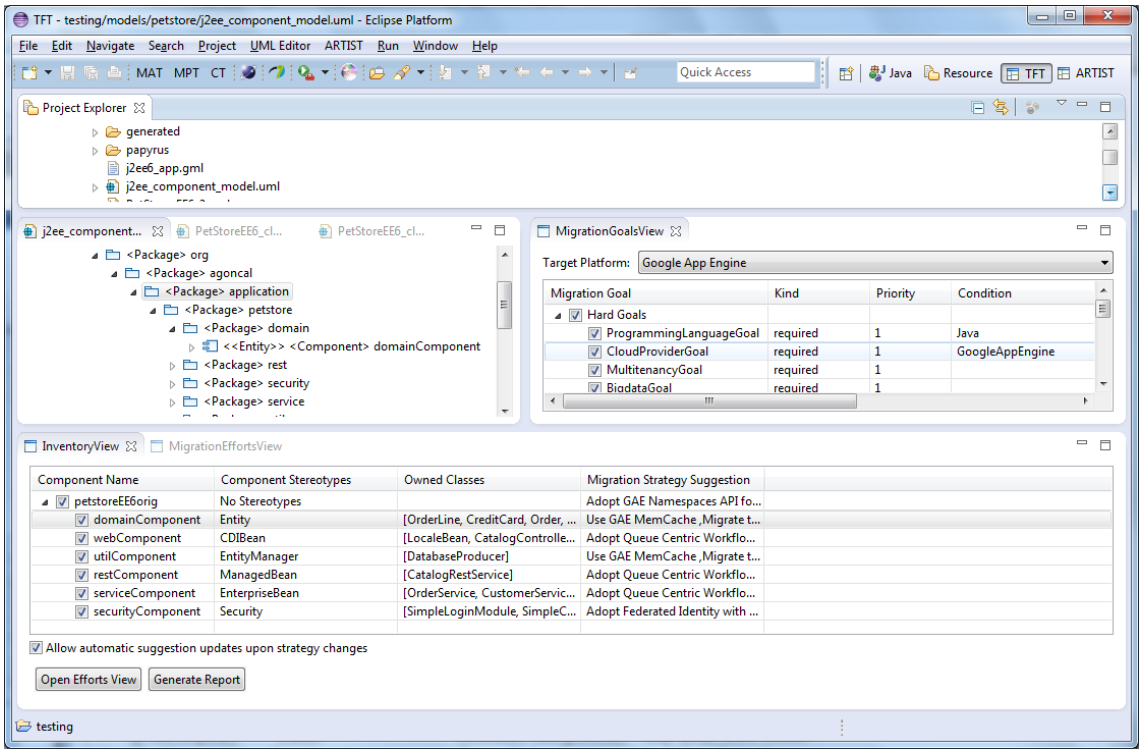


Figure 8 - Eclipse workbench with TFT views

At the top pane, in the project explorer a sample test project containing a JEE6 Component model diagram of the Java Petstore application is open. By right clicking the component model UML file (also Papyrus container and the DI files are supported), the Inventory view (bottom pane) can be opened. Also, j2ee6_app.gml can be seen inside the testing project. This MAT report file is used to open the Migration Goals View which can be seen on the middle right region of Figure 4: Eclipse workbench with TFT views.

On the middle-left pane, a UML Editor is open. The user is free to modify the model on this or another UML editor of her/his choice (e.g. Papyrus). The model with the new modifications is analysed and suggestions are updated once the user saves the changed model. On the right of this pane, the Migration Goals View can be seen. On the bottom pane, two tabs can be seen, Inventory View and the Migration Efforts View. These three views will be explained in detail in the following sections.

2.2.3.1 Migration goals view

To begin using TFT, the migration goals should be loaded which is done by showing a migration goals file in the Migration Goals View. In order to do this, the user should right click the GML file containing the migration goals which is generated by the MAT and select Show in Migration Goals View menu action.

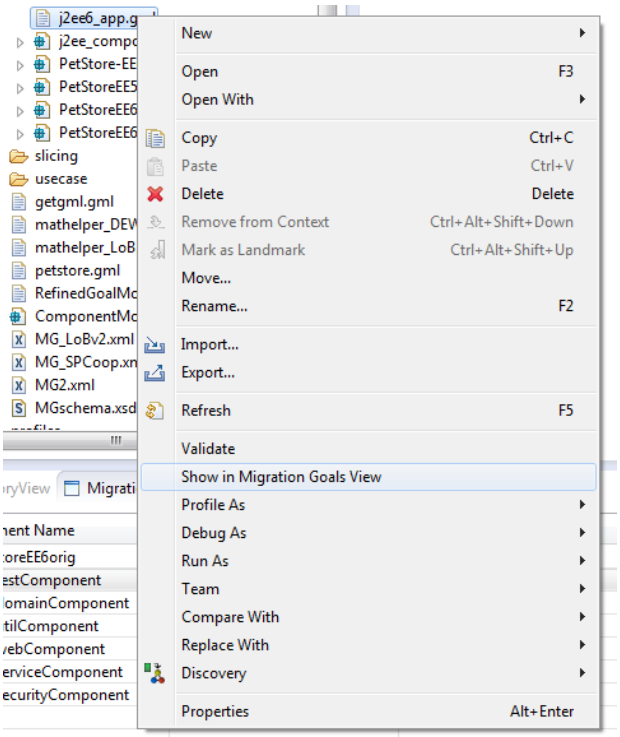
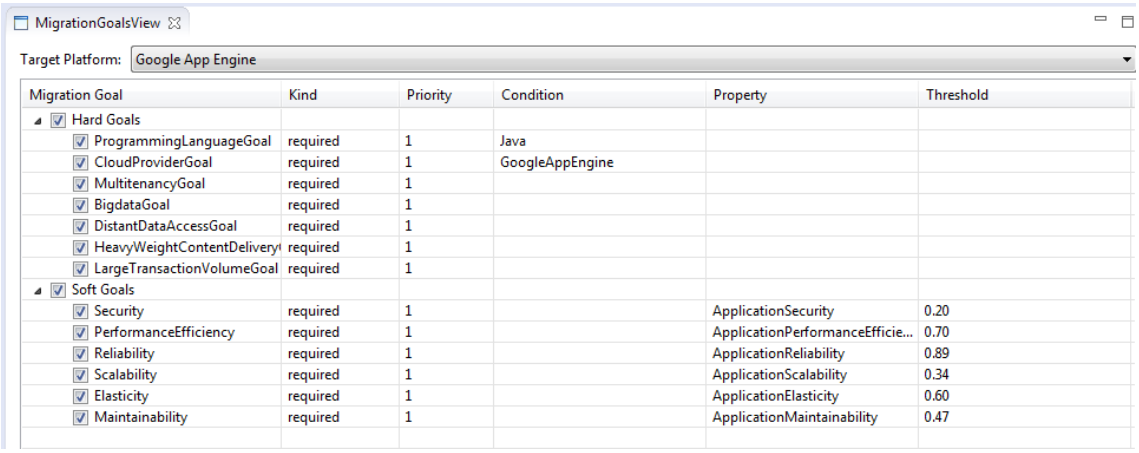


Figure 9 - Opening the migration goals view

The Migration Goals view consists of a combo box and a tree table.

- The combo box is filled with supported target cloud platforms with regards to the MAT report and default value is selected with respect to the value of the target platform in the migration goals. The combo box value can be changed by the user, which surely affects the suggested migration strategies in the Inventory View.

- The tree table lists the migration goals in two main categories; hard goals and soft goals, with their respective values. The user has no control over these values but can enable and disable the goals by clicking the checkboxes. In case the user wants to edit the values of the migration goals, he/she has to start over the MAT questionnaire again and go through the same process since these goals are determined by MAT based on the user's responses. Enabling/disabling the goals also affects the migration strategy suggestions in the Inventory View.



Migration Goal	Kind	Priority	Condition	Property	Threshold
Hard Goals					
ProgrammingLanguageGoal	required	1	Java		
CloudProviderGoal	required	1	GoogleAppEngine		
MultitenancyGoal	required	1			
BigdataGoal	required	1			
DistantDataAccessGoal	required	1			
HeavyWeightContentDelivery	required	1			
LargeTransactionVolumeGoal	required	1			
Soft Goals					
Security	required	1		ApplicationSecurity	0.20
PerformanceEfficiency	required	1		ApplicationPerformanceEfficie...	0.70
Reliability	required	1		ApplicationReliability	0.89
Scalability	required	1		ApplicationScalability	0.34
Elasticity	required	1		ApplicationElasticity	0.60
Maintainability	required	1		ApplicationMaintainability	0.47

Figure 10 - Migration goals view

2.2.3.2 *Inventory view*

The next step is to open the Inventory view. In order to open this view, the user must first open the Migration Goals View since the migration tasks to be suggested in the Inventory View depends on the migration goals. If the user tries to open the Inventory View before the Migration Goals View, a warning message pops up, informing the user about this, and asks for Migration Goals View to be opened first.

To open the inventory view of TFT plugin, the user should right click the component model UML file which is the output of model discovery process and select Show in Inventory View menu action. This action can also be found in the context menus of the files of type .di and Papyrus Model Containers.

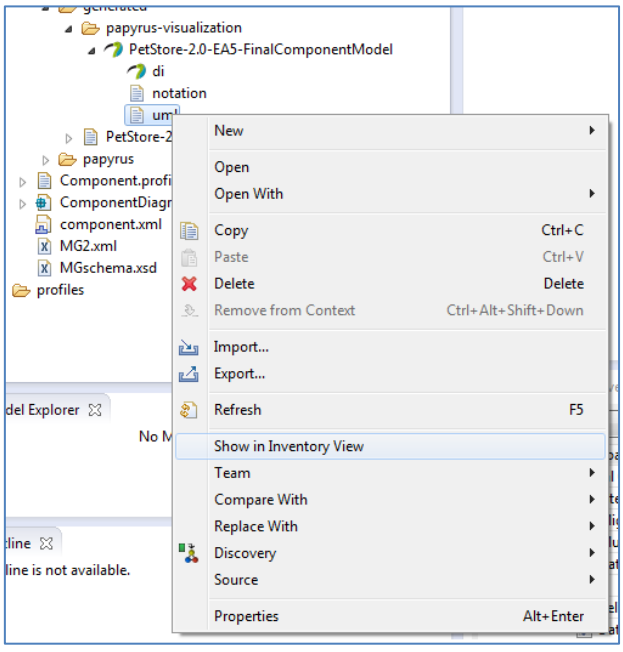


Figure 11 - Opening a model in inventory view

After selecting the “Show in Inventory View” menu option, the Inventory View will open.

The inventory view consists of a table which has the components of the selected component diagram. Component name, stereotypes, the classes owned by the component and the migration strategies suggested for the component by TFT can be seen in each row.

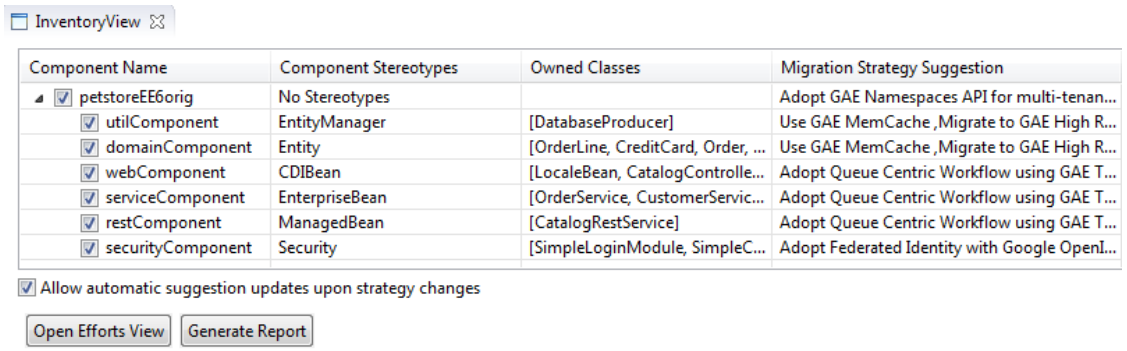


Figure 12 - Inventory view

The user is given the option to change the suggested migration strategy for a specific component. This is done by clicking the Migration Strategy Suggestion cell of the component in the table and clicking the ... button on the right-side of the cell. This action opens the Migration Strategy Selection Dialog which is described in detail in the next section.

Inventory view is fully responsive to the changes on the migration strategy selections and also to the changes on the component model. If the user modifies migration strategy selections for a component via the Migration Strategy Selection Dialog and clicks ok, the rule engine steps in and refreshes the migration strategy suggestions for all components. This automatic suggestion update can be disabled using the checkbox below the Inventory View table. Also, if the user decides to add a component, or changes a component's properties by using the component editor, the inventory view is refreshed and the strategy suggestions are recalculated when the changed model is saved.

As can be seen in **Figure 12 - Inventory view**, the report generation and opening the Migration Effort View functions are accessed through this view by the two buttons located at the bottom of the Inventory View.

2.2.3.3 Migration strategy selection dialog

The migration Strategy Selection Dialog opens when the user selects the migration strategy cell for a component in the Inventory View and clicks the ... button.

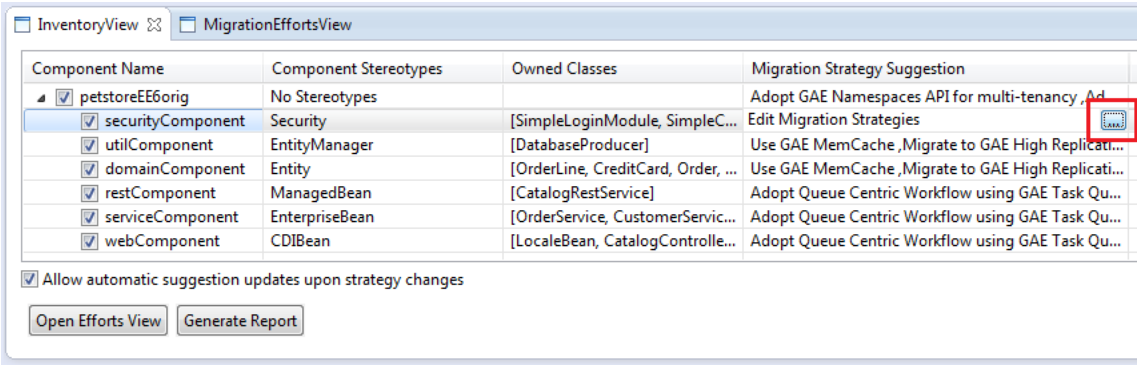


Figure 13 - Opening migration strategy selection dialog

In this dialog, the user is free to change the migration strategy suggested by TFT, remove strategies and/or add new ones.

If the selected migration strategies are modified, the user is warned about the possibility of the update on the migration strategy suggestions of the components that has a dependency relationship with the component selected, after the Ok button is clicked.

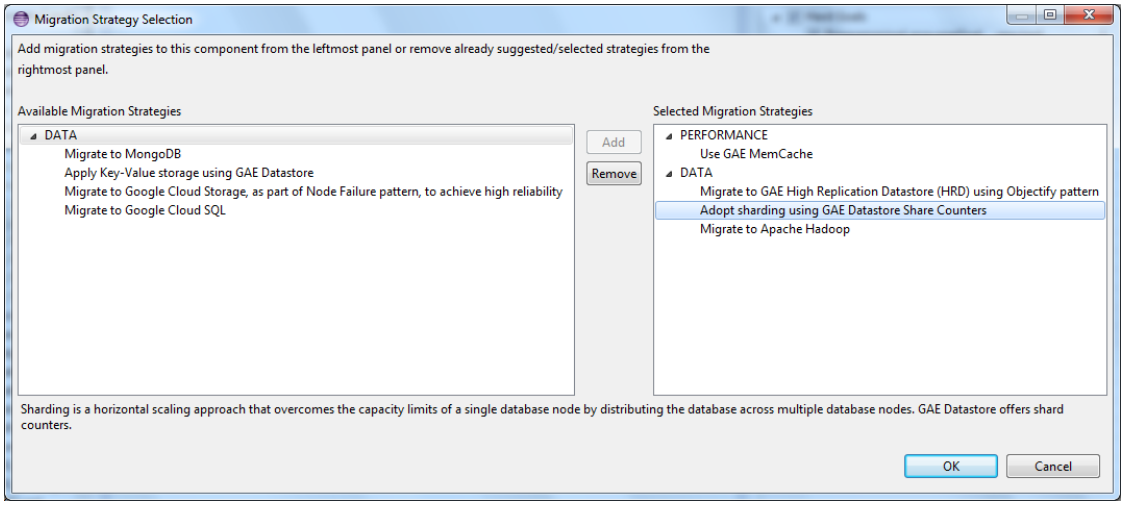

Migration Strategy Selection
Add migration strategies to this component from the leftmost panel or remove already suggested/selected strategies from the rightmost panel.
Available Migration Strategies
DATA
Migrate to MongoDB
Apply Key-Value storage using GAE Datastore
Migrate to Google Cloud Storage, as part of Node Failure pattern, to achieve high reliability
Migrate to Google Cloud SQL
Add
Remove
Selected Migration Strategies
PERFORMANCE
Use GAE MemCache
DATA
Migrate to GAE High Replication Datastore (HRD) using Objectify pattern
Adopt sharding using GAE Datastore Share Counters
Migrate to Apache Hadoop
Sharding is a horizontal scaling approach that overcomes the capacity limits of a single database node by distributing the database across multiple database nodes. GAE Datastore offers shard counters.
OK
Cancel

Figure 14 - Migration strategy selection dialog

2.2.3.4 Migration efforts view

To open the Migration Efforts View the user should click on the Open Efforts View button provided in the Inventory View. At this point, the user will see the dialog below asking the source file location(s) of the selected component model.

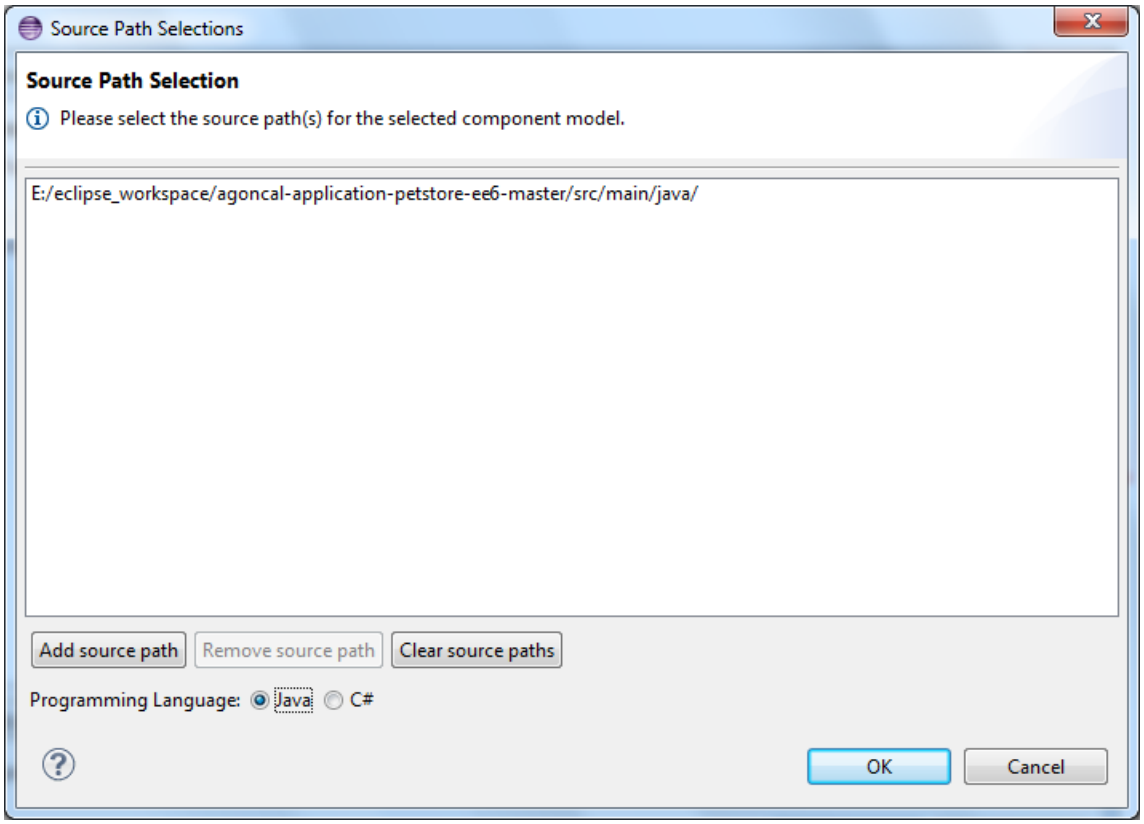
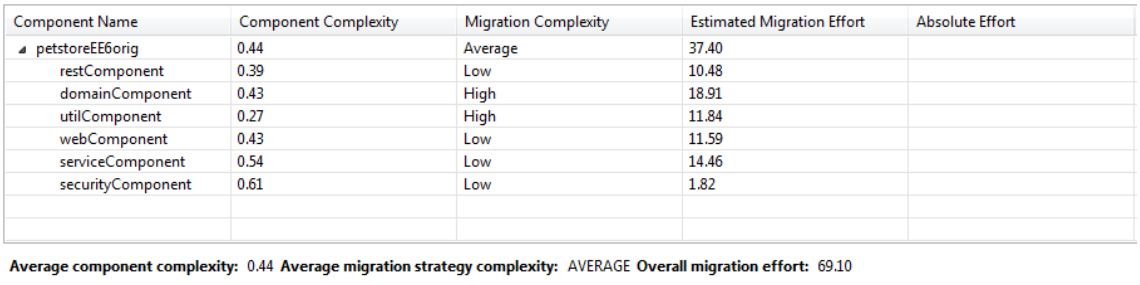


Figure 15 - Source paths selections dialog

Using this dialog's "Add source path" button the user should locate the source folders containing the classes referred by the component model, select the programming language then press OK to proceed and open the Migration Efforts View.

This view contains a table where for each component, the component complexity, migration complexity, overall migration effort and absolute effort is shown (**Figure 16 - Migration efforts view**). Overall averages can also be seen at the bottom of the view. The absolute effort values are calculated upon user interaction within the dialog which the user can double click a component to bring forth. In this dialog, complexity, effort and absolute effort information for each individual migration strategy suggested for this component can be seen.

The user is free to fill in the absolute value for a strategy of his/her choice with a man-day value and let TFT estimate the man-day values for the rest of the strategies and also for other components. The user can also fine tune the estimated absolute effort values after this automated calculation.



Component Name	Component Complexity	Migration Complexity	Estimated Migration Effort	Absolute Effort
petstoreEE6orig	0.44	Average	37.40	
restComponent	0.39	Low	10.48	
domainComponent	0.43	High	18.91	
utilComponent	0.27	High	11.84	
webComponent	0.43	Low	11.59	
serviceComponent	0.54	Low	14.46	
securityComponent	0.61	Low	1.82	

Average component complexity: 0.44 **Average migration strategy complexity:** AVERAGE **Overall migration effort:** 69.10

Figure 16 - Migration efforts view

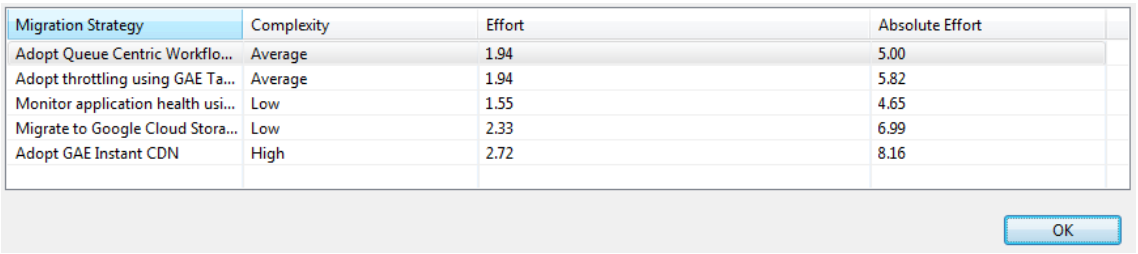


Figure 17 - Effort details dialog

2.2.4 Licensing information

This component is offered under EPL v1.0 license.

2.2.5 Download

Source code for this version of the TFT prototype may be viewed\downloaded from the ARTIST Tooling Github repository, here.

3 SCC

3.1 Implementation

3.1.1 Functional Description

The Software Complexity Component (SCC) aims to provide set of metrics in the technical feasibility phase prior to the migration. Thus, the results from metrics analysis can provide a better insight whether migration should be feasible or not, supporting the estimation of the effort required for performing the migration project.
In the first version of the prototype delivered in M12 [7], the Level of Complexity (LOC) metric was provided for each Java class. In the second version of the prototype, at M24, the Modifiability, Understandability and Scalability (partially) metrics were calculated [5], both for Java and C# programming languages.
The current version of the prototype all the metrics needed to calculate the final Maintainability metric are provided:

$$Maintenance = 2.399 + 0.493 \times Modifiability + 0.474 \times Understability + 0.524 \times Scalability + 0.507 * LOC$$

Where:

$$Modifiability = 0.629 + 0.471 \times NC - 0.173 \times NGen - 0.616 \times NAggH - 0.696 \times NGenH + 0.396 \times MaxDIT$$

$$Understability = 1.66 + 0.256 \times NC - 0.394 \times NGenH$$

$$Scalability = 0.182 \times 0.99 \times AC + 0.100 \times EC + 0.097 \times ND - 0.036 \times PC + 0.068 \times DMS$$

$$LOC = 0.269 + 0.008 \times Coupling + 0.181 \times cohesion + 0.119 \times CC + 0.084 \times ILCC$$

These metrics are calculated per component (defined in the component model by the MUT).

Apart from the Maintainability metric, and the intermediate metrics required to calculate it, a new metric related to the reusability of the code is provided for the SCC. The reusability metric is calculated by the ARTIST MAT (Maturity Assessment Tool)[1].The SCC calls the MAT in order to obtain the percentage of reusable lines of code and reusable classes. These metrics will also support the calculation of the required effort. Two reusability metrics are calculated:

- Reusability lines: this is the percentage of the code lines that are reusable.

Reusability methods: this is the percentage of the methods that are reusable

3.1.1.1 Fitting into overall ARTIST solution

SCC prototype is part of the Technical Feasibility Tool. In the first version of the prototype, SCC and TFT were provided as separate components, but in the current version of the prototype, they are provided as a single tool.

TFT (and thus SCC) is the tool that ARTIST project provides to perform the Technical Feasibility analysis proposed in pre-migration phase of the ARTIST methodology (see **Figure 1 - ARTIST Migration Feasibility Assessment Package**).

3.1.2 Technical description

This section describes the technical details of the implemented software.

3.1.2.1 Prototype Architecture

The current SCC prototype architecture is a Java API that explores source files an UML models to generate several metrics of a specific project. The following image depicts the overall architecture:

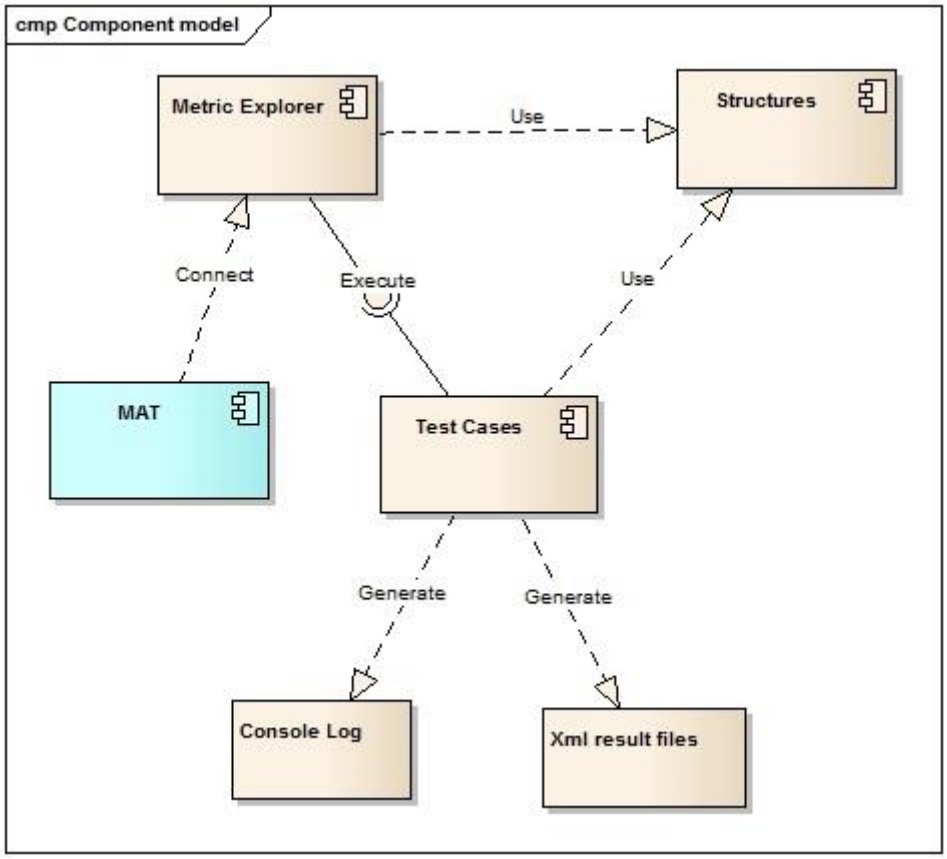


Figure 18 - SCC high level architecture

While the main objective of the Artist Metrics Generator is to expose an API that any other plug-in or RCP could use to obtain the metrics generated in the Artist project, it also provides Test Cases to access the same functionality as if used programmatically. The generated metrics are available in XML files and console log.

3.1.2.2 Components description

The current SCC prototype component comprises three components:

- Metric Explorer: This is the main component of SCC current prototype. It provides the calculation of all the required metrics that are used to generate the new ARTIST metrics. Besides, it also provides exporting features to convenient formats like XML or JSON. Metric Explorer also includes a connector to the MAT (Maturity assessment tool). This connector allows the user to generate reusability metrics.
- Structures: This component contains the structures of the inputs and outputs models that the Metric Explorer uses. It also provides the functionality for generating the output file formats (XML, JSON).

- Test Cases: This component is provided for implementing the testing of the Metric Explorer component. It generates several use cases that test the functionality of the SCC. The test case generates console logs and XML files with several examples (DEWS and JavaPetStore).
- MAT: Is the maturity assessment tool. SCC connects the MAT tools to calculate the Reusability metrics.

3.1.2.3 Technical specifications

All the components are developed in JavaSE 1.6. So this is the minimum java version for executing the API. There are no any other requirements.

User Interface

There is no user interface implemented as the result obtained from SCC is to be consumed by TFT-TFT and used for the required effort calculation (see section 4). For executing the Metrics Explorer API the user has to execute the test cases included in the API. Several input parameters can be changed for obtaining new metrics of different projects.

Back-end

There is no persistency implemented neither planned for this API. Thus, every time the new metrics are needed, they have to be calculated. Persistency is delegated to API consumers.

3.2 Delivery and usage

3.2.1 Package information

The following image depicts the package structure of the main component, the Artist metrics generator plug-in.

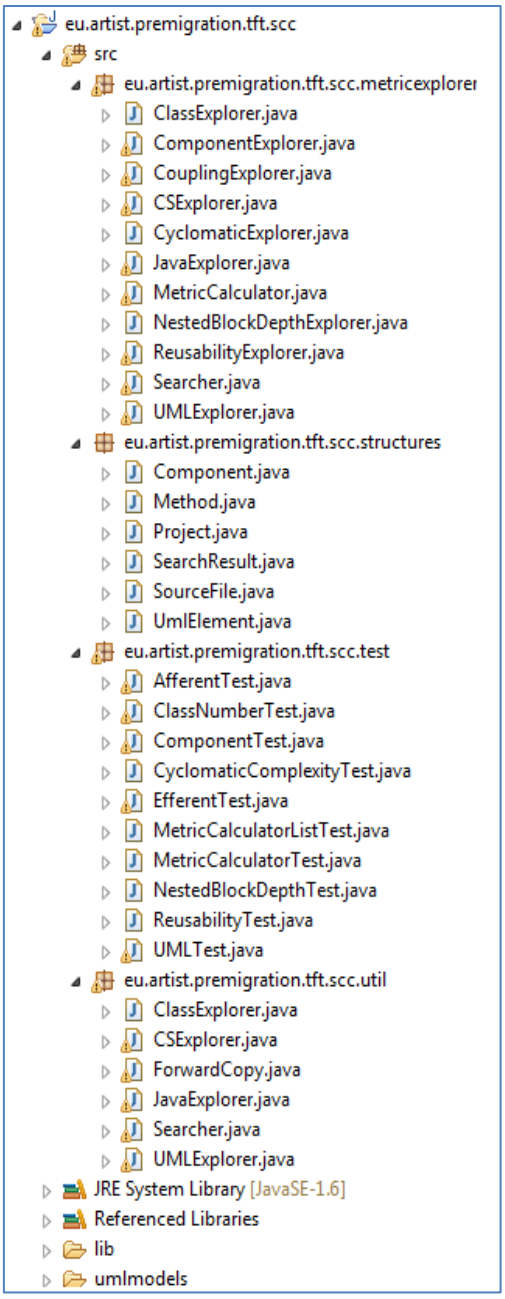


Figure 19 - Package structure of SCC

- **eu.artist.premigration.tft.scc.metricexplorer**: Contains the classes for exploring the UML models and the source code.
- **eu.artist.premigration.tft.scc.strucctures**: Contains the classes of the structures used by the metric explorer component.
- **eu.artist.premigration.tft.scc.test**: Contains the test cases classes for executing the metric explorer component.
- **umlmodels**: This folder contains the UML models used in the test cases and other UML models also.

3.2.2 Installation instructions

In this version of the prototype this plug-in requires manual installation. The user has to import the components to the Eclipse workspace manually.

3.2.2.1 Requirements

All the components are developed in JavaSE 1.6. So this is the minimum Java version for executing the API.

3.2.3 User manual

Import the project into the Eclipse workspace:

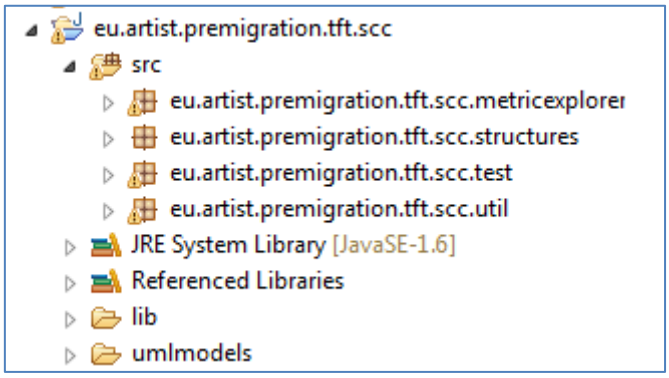


Figure 20 - SCC Project

Open the eu.artist.premigration.tft.scc.test package

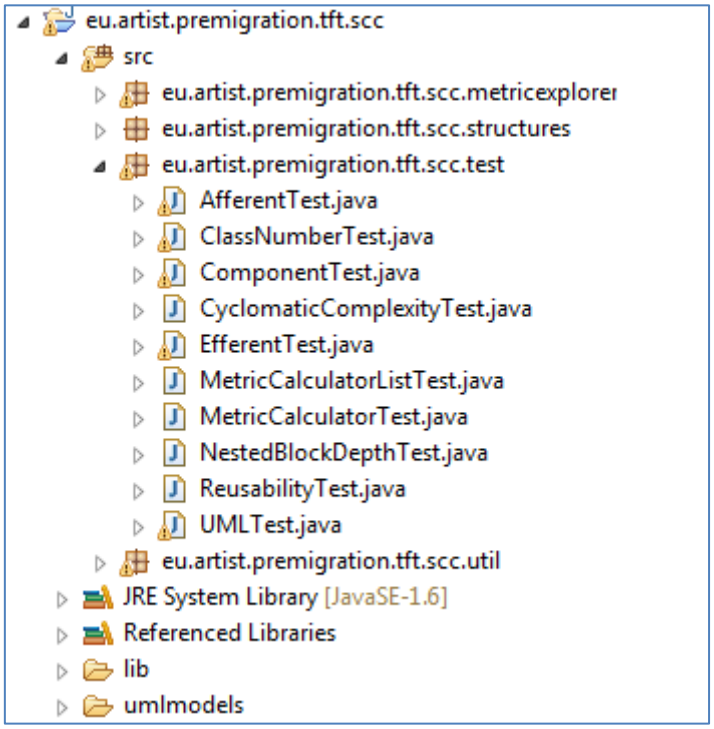


Figure 21 - Test package

Right click in a test case and select the Run as Java Application option:

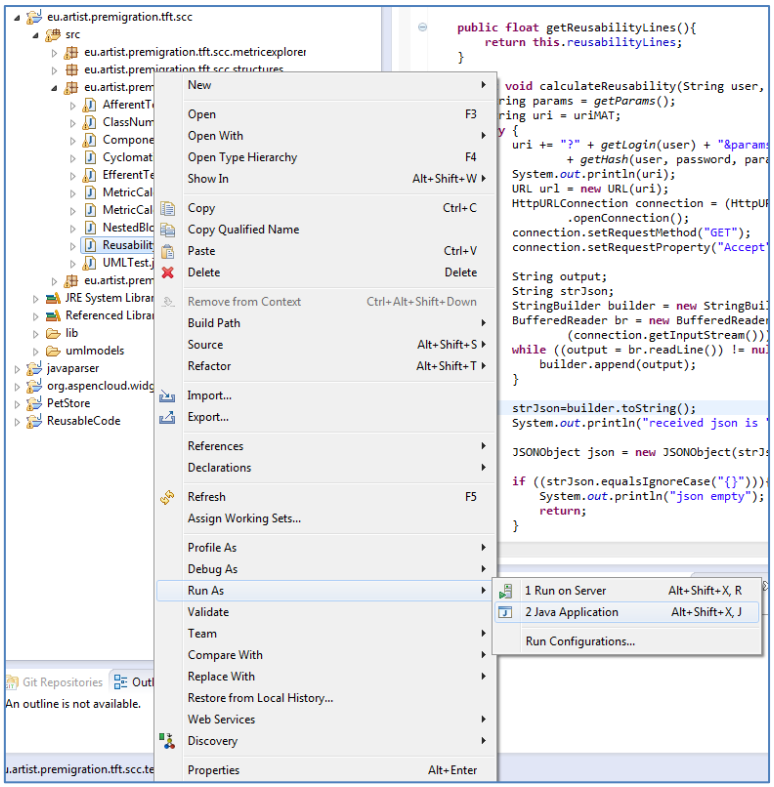


Figure 22 - SCC testing

***Note**: The user has to change the "hardcoded" input parameters of the test cases manually.*

3.2.4 Licensing information

This component is offered under EPL license.

3.2.5 Download

This second release of SCC is available in the ARTIST Github, more precisely at the following links: binary and source.

4 Interaction of TFT components

The two components of TFT interact with each other in order to present the user an effort estimation of the selected migration strategies for each component. Figure 4 depicts an overall picture of this interaction.

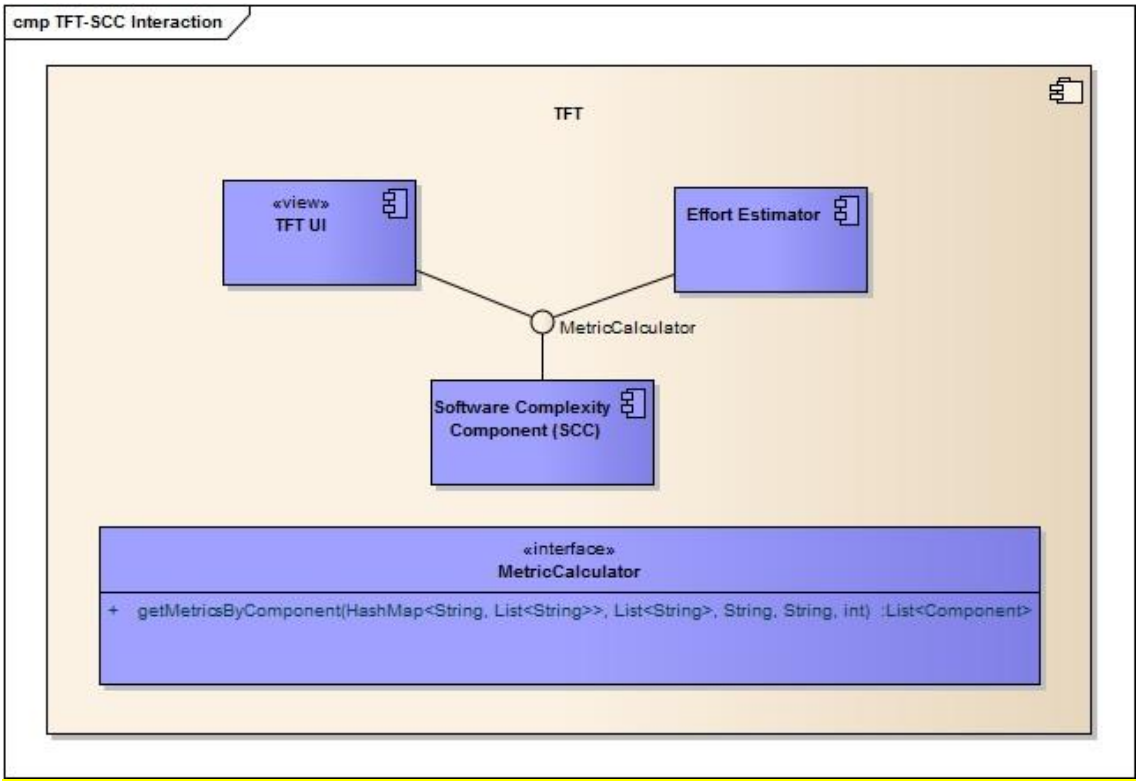


Figure 23 - Interaction of TFT components

SCC offers the MetricCalculator class containing the *getMetricsByComponent* method to be used by TFT to obtain the metrics calculated by SCC. The method signature can be found below.

- `getMetricsByComponent(HashMap<String, List<String>> classFiles, List<String> srcPaths, String ComponentModelPath, String ClassModelPath, int type)`

This method requires a map which has the component name as the key and the class files contained by this component as the value, a list of source paths which the source files can be found in, the path of the component model, the path of the class model and the source type (i.e. Java, C# etc.). The method returns a list of components (`eu.artist.premigration.tft.scc.structures.Component`), each containing the metric information of the TFT components that are included in the component model of the migration candidate (`eu.artist.premigration.tft.tft.model.Component`).

TFT uses the metric information and the migration strategy complexities to calculate the estimated effort for each component's migration towards cloud. The calculated effort estimation is displayed in the Migration Efforts View which can be accessed from the Inventory View.

5 Conclusions

This document described the functionality, architecture, delivery method and installation/usage details of Technical Feasibility Tools, detailed the SCC component and the interaction between TFT components. The third version of TFT prototype containing several new features has also been provided.

In this version of the deliverable, SCC component introduced a new version of the Maintainability metric and also a new metric called Reusability metric which contribute to increase the accuracy of the estimation for the migration strategy efforts. An Eclipse perspective and a preference page for TFT have been introduced to further improve the usability of the tool. With the introduction of GML MAT report support, improvements in performance and the new functionalities in the Migration Efforts View, TFT has been updated to be more informative and responsive to the user. The knowledge base of TFT has also been enlarged with new rules and new migration strategies derived from work package 9 [4].

In the next iteration, the focus will be on enhancing the knowledge base of rules supporting more complex scenarios and more specialized migration strategies, making use of the unused data provided in the GML report to enhance the rules and the implementation of the TFT repository to store the historical data to be used by TFT itself.

6 References

[1] D5.1.1 – Specification of the Business and Technical Modernization Assessment in ARTIST

[2] D6.2.1 – ARTIST – Methodology

[3] D8.3 – ARTIST - D8.3 Methodology and techniques for model understanding M18

[4] WD9.4 – ARTIST – WD9.4 Collection of Optimization Patterns

[5] D5.3.2 - Technical Feasibility Tools

[6] D6.4.1 - ARTIST Integrated Architecture

[7] D5.3.1 – Technical Feasibility Tools

7 Appendix: TFT knowledge base (library of rules)

7.1 Migration Strategy Taxonomy

In order to migrate a system to the cloud, certain tasks have to be fulfilled. These tasks may require modifications on the system itself and definitely some work has to be done on the target platform as well. These tasks can be organized within 4 main categories, Installation & Configuration, Modification, Migration and finally, Testing, which consist of more precise, sub-categories.

Installation and Configuration

Software installation and/or configuration may be needed locally or on cloud, for IaaS migrations, depending on the candidate system for migration before starting the code modification. The installations and configurations may be required or may just be a case of "nice to have".

An example for a requirement would be the case where a Java application, which uses JPA, is to be migrated to the cloud with SimpleDB. In this situation a third party library, SimpleJPA has to be installed and configured.

Although packaging and deployment of the application are in the context of the Migration category, some installation and/or configuration might be needed to automate/ease some parts of the migration process. For example: Azure plugin for Visual Studio must be installed in order to benefit from its automated packaging feature.

Modification

Modification category focuses on the required modifications on the system before migration and consists of 3 sub-categories: Database Changes, Code Changes and Connection Changes. The scope of modification depends on many parameters; target platform, the modules that are planned to be migrated or not, compatibility of the local and the target platforms etc.

- **Database changes:** If the database is also planned to be migrated to the cloud, it is very likely that some modification will be needed. Database scripts and schema modifications might be necessary in following situations:
    - Difference of database versions
    - Difference of database variants (MySQL vs MSSQL)
    - Difference of database types (Relational vs NoSQL)
- **Code changes:** If migration goals include enhancing overall performance, switching to more contemporary technologies etc., changes on the written code would be inevitable. Also if there has been a modification on the database, code change would probably be needed.
    - Database access layers, models may need to be rewritten. An example would be JPA to SimpleJPA conversion.
    - In a case of an MVC application migration, CDI Managed beans and controllers might be modified to integrate with Spring.

- **Connection changes:** Typically, two components of a system are connected with a LAN connection before the migration. According to the migration strategy, this connection may change to a WAN connection or a LAN connection in cloud. For either case, modifications below might be necessary,
    - Protocol optimizations to ensure data transfer efficiency
    - Security optimizations. If there will be a switch from LAN to WAN, some additional threat scenarios should be considered.

Migration

Being the last step before testing, this category defines the process of configuration and preparation of the modules targeted for migration and should be trivial if the previous steps are handled properly.

- **Configuration:** Application configuration files should be change\modified to handle the modifications done on the code in the previous category and also according to the target platform requirements.

  Additionally, some configurations on the project nature might be necessary in order to make use of some third party plugins. For example, Azure plugin for Visual Studio automates package and configuration file creation process, but in order to use the plugin, the project to be migrated must be a “Web Application Project”. If it is a “Web Site Project”, it has to be converted to a Web Application Project first.

- **Preparation:** In order to deploy the application and/or the database to the cloud some certain preparations may be required.
    - Preparation of the database: If the scripts had not been changed in the code modification step, some configuration changes might be necessary in order to use third party tools for DB migration automation.
    - Preparation of the application: Before the deployment, the application has to be packaged. In case of migrating to PaaS application may be required to be packaged according to a certain format.

Testing

This one of the most important - if not the most important - part of the migration as a whole. A successful deployment to cloud (migration process) is a proof of some of the actions taken are not erroneous; configuration changes, installations on IaaS etc. but this does not ensure that the application would run as expected.

Testing process should be well planned and carried out carefully. One example procedure for testing might be as follows:

- Testing if database migration to cloud is successful.
- Testing the application in local servers can communicate with the DB in Cloud.
- Testing if the migration of the application to cloud is successful
- Testing if the application in cloud can communicate with the DB in cloud.

- Carrying out functional, performance, security etc. tests

7.2 Migration strategies

Adopt Cloud Watch: This migration strategy suggests adopting the Cloud Watch Service offered by Amazon EC2. Amazon CloudWatch is a monitoring service offered by Amazon that controls all the time the use of cloud infrastructure resources and allows system administrators to establish scaling rules based on the values of certain metrics.

Migrate to App Engine Datastore: This strategy suggests that necessary changes should be made on the component so that it becomes compatible with Google App Engine's Datastore. This service provides a schema-less object datastore including a query engine, atomic transactions support via the Datastore API. Its primary data repository (High Replication Datastore) replicates its stored data in multiple data centers providing high read/write availability and durability. Datastore is scalable in terms of reads and writes. Writes scale by distributing write data when necessary while reads scale by only permitting queries whose performance depends solely on the result set. Query serving uses pre-built indexes which impose restrictions on the supported operations over the stored data.

Migrate to Task Queue API: This strategy suggests that the related component should adopt and use the Google App Engine's Task Queue API (TQA). TQA allows applications to perform work outside of a user request, initiated by a user request. The work is divided into separate tasks and these tasks are pushed to task queues in order to be processed in aggregate.

Adopt Amazon Direct Connect: This migration strategy suggests adopting Amazon Direct Connect. This service offered by Amazon, enables the establishment of a dedicated network connection between AWS and other resources. It serves for keeping constant network latency needed in case of real-time data feeds as well as for utilize Amazon's infrastructure together with other infrastructure controlled by the user.

Test Database Connections: This migration strategy suggests testing the database connections after migration. Testing of the connection between the local servers and the DB in the cloud and the connection between the application in the cloud and the DB in the cloud should be carried out.

Adopt Google Search: This migration strategy suggests adopting the Google Search to be used in the migrated application. This service offers the ability of performing search over structured data.